This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

Privacy-Aware Caching in Information-Centric Networking

Gergely Acs, *Member, IEEE,* Mauro Conti, *Senior Member, IEEE,* Paolo Gasti, *Member, IEEE,* Cesar Ghali, *Member, IEEE,* Gene Tsudik, *Fellow, IEEE,* Christopher A. Wood, *Member, IEEE*

Abstract—Information-Centric Networking (ICN) is an emerging networking paradigm where named and routable data (content) is the focal point. Users send explicit requests (interests) which specify content by name, and the network handles routing these interests to some entity capable of satisfying them with the appropriate data response (producer). One key feature of ICN is opportunistic in-network content caching. This property facilitates efficient content distribution by reducing bandwidth consumption, lessening network congestion, and improving the content retrieval latency by users (consumers). Unfortunately, the same feature is also detrimental to privacy of content consumers and producers. Simple to implement, and difficult to detect, timing attacks can exploit ICN routers as "oracles" and allow an adversary to learn whether a nearby consumer recently requested certain content. The attack leverages a timing side channel that relies on router caches and is implemented by requesting a few packets from each piece of content being probed. Similarly, probing attacks that target content producers can be used to discover whether certain content has been recently distributed. After analyzing the scope and feasibility of such attacks, we propose and evaluate some efficient countermeasures that offer quantifiable privacy guarantees while retaining the benefits of ICN.

Index Terms—information-centric networking (ICN), named-data networking (NDN), content-centric networking (CCN), cache privacy, timing attacks.

1 Introduction

Today's Internet has become a *de facto* public utility with a large percentage of the world's population relying on it for numerous activities. However, despite its unparalleled success and popularity, the current IP-based architecture is rapidly aging. As a consequence, a number of research efforts [30] have recently started in preparation for the next-generation Internet architecture.

One key motivator for a new Internet architecture is the fundamental shift in the nature of traffic. What was once mainly low-bandwidth interactive (e.g., remote log-in) and store-and-forward (e.g., email) communication in the early days of the Internet is now web- and content-dominated traffic. At the same time, massive and ever-increasing amounts of content continue to be produced and consumed (distributed) over the network. This phenomena is manifested over file sharing services such as Dropbox and media sites such as YouTube and Netflix. In summary, communication in the Internet has shifted from a telephony-like conversation between two IP interfaces towards a consumer who wants content and wants it fast, regardless of where it comes from. This motivates reconsidering the current Internet architecture and exploring alternatives.

Information-Centric Networking (ICN) is an emerging network architecture in which the focal point is named and routable data (content), rather than hosts and addresses. In ICN, a consumer requests content by *name* (i.e., expresses *interests* for the content) and the network takes care of finding and returning the data. The ICN approach moves hosts into the background by treating named content as a first-class object. One important ICN feature is opportunistic in-network content caching. Its goal is to reduce congestion while improving throughput and latency for popular content. In contrast to IP, ICN routers can often satisfy interests using previously forwarded content. Consequently, content might be served from a router's cache rather than its original producer.

Despite its obvious benefits, content caching in ICN raises a major privacy issue that we summarize below and then discuss, in more detail, in Section 3. Suppose an adversary Adv wants to determine whether a consumer (Alice) recently requested certain content C. Assume Adv and Alice share a first-hop ICN router R, e.g., they are neighbors who are served by the same ISP, and Adv can measure the round-trip time (RTT) to R (Adv $\leftrightarrow$ R). Adv issues an interest for C and measures the corresponding RTT. By comparing the expected and actual RTTs after retrieving C, Adv can determine whether C was retrieved from R's cache, which indicates that Alice previously requested it.[1] Specifically, if two RTTs are approximately equal, then Adv can conclude that C must have been served by R.

Similarly, suppose that Adv wants to learn whether a content producer (Bob) has been recently asked for, and has subsequently distributed, C. Assuming that Adv and Bob are separated by at least one router, Adv measures or estimates the RTT between itself and Bob and then requests C. If the latter RTT is lower than the former, Adv concludes that C was fetched from some ICN router cache and, therefore, at least one consumer recently requested it. Furthermore, a combination of these two attacks can be used to learn whether two parties (Alice and Bob) have been recently, or still are, involved in two-way interactive communication, e.g., voice communication or SSH.

These attacks do not require Adv to have any special privileges: the interaction between Adv and ICN routers is normal. It might appear that Adv can learn the same information by simply eavesdropping on Alice or Bob. However, eavesdropping

- *M. Conti was supported by Marie Curie Fellowship PCIG11-GA-2012-321980, the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), the EU-India REACH Project (agreement ICI+/2014/342-896), and by the project "Content Centric Networking: Security and Privacy Issues" funded by the University of Padua.*
- *C. Ghali, G. Tsudik, C. A. Wood are with the Department of Computer Science, University of California, Irvine, CA, 92697.*
- *P. Gasti, C. Ghali and G. Tsudik were supported by the NSF under project CNS-1040802 – "FIA: Collaborative Research: Named Data Networking (NDN)."*
- *P. Gasti performed work in part while at UC Irvine. C. A. Wood is supported by the NSF Graduate Research Fellowship DGE-1321846.*

1. Clearly, there might be other users, besides Alice and Adv. However, this would not change the nature of the attack.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

requires colocation with Adv since it must also be physically near the victim (e.g., the same ethernet segment, wired or wireless). In contrast, the aforementioned attacks require neither real-time presence nor proximity. Moreover, using encrypted links between consumers and their first-hop routers prevents Adv from successfully eavesdropping on their traffic.

At first glance, ICN seems inherently safe against cache privacy attacks: if $k > 1$ consumers share the same router's cache, Adv cannot determine exactly *which* or *how many* requested particular content (if the content does not expose consumer-identifying information). Thus, consumers seem to benefit from some form of k-anonymity. However, for many types of traffic, e.g., email, instant messaging, and voice, consumer identities can be determined from the content or its name.[2] Moreover, even k-anonymity may be insufficient if Adv possesses any auxiliary information about neighboring consumers, or it simply wants to determine whether *any* consumer requested a particular piece of content.

Note that content encryption is not sufficient to mitigate these attacks. Encryption conceals the payload of a content object and, optionally, part of the name. However, names cannot be fully encrypted, since doing so would prevent content objects from being routed.[3]

Based the above discussion, there is an inherent conflict between utility of in-network caching and privacy for consumers and producers. The resulting privacy problem and potential countermeasures are the subject of this paper. We believe it is imperative to address or at least mitigate them before ICN can be deployed on a large scale.

To the best of our knowledge, we were the first in addressing these problems with our previous work in [3], for which the current paper is an extended version. In particular, in the current paper, we extend [3] with the additional following contributions:

- Elaboration upon the efficacy of cache timing attacks in CCN and NDN – two types of ICN architectures.
- Extension of the cache privacy attack to local and distributed adversaries.
- Complete proofs for the privacy-preserving cache mechanisms.
- A protocol to improve the efficiency of privacy-preserving caches in trusted domains.

Organization. Section 2 overviews ICN. We show how to perform privacy attacks, and evaluate their effectiveness with experiments run on the current NDN testbed [28], in Section 3. Our adversary and privacy models are described in Section 4. Next, sections 5 discusses proposed countermeasures for local adversaries. Distributed (non-local) adversaries are covered in Section 6. Considerations for both scenarios are discussed in Section 7. Improvements to our countermeasures that aim at reducing the delay of content delivery are explored in Section 8. Experimental evaluation and real-world impact of our countermeasures are presented in Section 9. An optimization that further reduces the overhead of our countermeasures and improves the end-user experience is proposed in Section 10. Section 11 reviews related work. Finally, we present our conclusions and future directions in Section 12.

2. In practice, it is likely that this type of interactive or consumer-specific traffic would be encrypted end-to-end, thereby rending caching useless. We discuss this in more detail in Section 5.1.

3. Onion routing over Named-Data Networking (NDN) – an instance of ICN – has been investigated in [10] and [34]. These approaches allow parties to encrypt content names. However, they both incur a non-negligible performance burden.

2 ICN Essentials

This section provides an overview of the essential parts of ICN by explaining Named-Data Networking (NDN) [27] and Content-Centric Networking (CCN) [1], both of which are derivatives of the general ICN architecture. Given familiarity with the subject and both of these variants, this section can be skipped without loss of continuity.

2.1 NDN Overview

NDN [27], [37], [19] is one of the National Science Foundation's Future Internet Architecture (FIA) projects[4], is a network architecture based on named content. Rather than addressing content by location, NDN refers to it by human-readable names. A content name is composed of one or more variable-length components that are opaque to the network. Component boundaries are explicitly delimited by "`/`" in the usual URI representation. For example, the name of a BBC news content for October 10, 2015 might look like: `/bbc/news/2015oct10`. Large pieces of content must be split into chunks or fragments. For example, chunk 263 of a YouTube video published by Alice could be named: `/youtube/alice/concert.avi/263`. Since NDN's main abstraction is content, it is not possible to directly address "nodes" (routers), although their existence is assumed.

NDN communication adheres to the *pull* model: data is delivered to consumers only upon explicit request. A consumer requests content by sending an *interest* with the name of the desired content. If an entity such as a router or a host forwarder can "satisfy" a given interest it returns the corresponding *content*. A content named N is never forwarded or routed unless it is preceded by an interest for name N. Strictly speaking, a content named $N' \neq N$ is considered a match for – and therefore can be sent in response to – an interest for N if and only if N is a prefix of N', e.g., `/BBC/news/2015oct10` matches `/BBC/news/`. In other words, content name matching in NDN is based on longest-prefix matching. Interest and content are the only types of packets in NDN.

According to the current specifications [2], interest message packets contain several delegated fields. In this paper we are only interested in the following:

- `Name` – NDN name of the requested content.
- `Scope` – specifies how far the interest message will be transmitted. `Scope` can take one of these values:
  - 0 – do not propagate beyond the local NDN daemon,
  - 1 – only propagate to the application layer of the current NDN node,
  - and 2 – do not propagate beyond the next hop node. This guarantees that interests will either be satisfied or dropped by the NDN node that is only one hop away.
- `Exclude` – contains information about name components that **must not** appear in the name of a returned content. This field can be used to exclude one or more content objects by referring to their hash, which as noted above, is considered to be an implicit last component of each content name.

Content object packets also have a `Name` and `Payload` field. We will describe the use of these later.

All NDN nodes (consumers, routers, and producers) maintain the following three components [8]:

4. In today's Internet, security and privacy problems were (and are still being) identified along the way and patches are retrofitted; some haphazardly. In contrast, the NSF FIA program stresses early awareness as well as support for features and counter-measures, from the outset. To this end, security and privacy issues in all FIA projects, including NDN, are being investigated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

- *Content Store* (CS) – A cache used for content caching and retrieval. (From here on, we use the terms *CS* and *cache* interchangeably.)
- *Forwarding Interest Base* (FIB) – A routing table that maps name prefixes to outbound network interfaces.
- *Pending Interest Table* (PIT) – A table that maps currently not-yet-satisfied (i.e., pending) interests to sets of corresponding inbound interfaces. Values (entries) in the PIT also store additional metadata (see [14]), but that is not useful for this work.

When a router receives an interest for name N, and there are no pending interests for a matching name in its PIT, it forwards the interest to the next hop(s) according to its FIB. For each forwarded interest, a router stores the interest name N and the interface on which it arrived in the PIT. However, if an interest for N arrives while there is already an entry for the same content name in the PIT, the router collapses the present interest (and any subsequent ones for N) and stores only the interface on which it was received. When content is returned, the router forwards it out on all interfaces on which an interest for N has arrived and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry a "source address".

Each NDN router can have a cache whose size is determined by local resource availability. Routers unilaterally determine what content to cache (if any) and for how long. Whenever an interest is received, a router first checks whether it can be satisfied from its cache. Therefore, NDN also lacks the notion of "destination address" since content objects can be served from any node (router or host) in the network. For this reason, most content objects carry a digital signature created by the original producer[5] in order for consumers (and in some cases routers [14]) to verify the authenticity of content, regardless of which entity actually served it in the network.

2.2 CCN Overview

CCN [1] is a research project lead by PARC. In general, the CCN and NDN protocols are quite similar.[6] However, some differences between the two are important from a privacy perspective. We highlight some of the distinct CCN properties relevant to this paper below.

- CCN content matching is based on an exact match. A named content satisfying an interest must include the exact same name in the corresponding interest.
- There are no `Scope` or `Exclude` fields in interest messages.

Enforcing exact content name matching and removing the `Scope` and `Exclude` fields eliminates one critical privacy attack we discuss in the next section.

3 Cache Privacy Attacks

In this section we describe the cache privacy attacks mentioned in Section 1. Our goal is to show that Adv can learn whether specific content C was recently requested by consumer Cr (Alice) by probing R's cache. To do so, Adv issues an interest for C and measures the delay τ_1 required to retrieve it. It then picks another (existing) content C' and requests it twice in rapid succession. The first time, C' might be fetched from its original producer P or from R's cache. However, the second time, C' is fetched from R's cache with high probability. Let τ_2 denote the retrieval delay for the latter. If $\tau_1 \approx \tau_2$, Adv concludes that Cr recently requested C. Otherwise, if $\tau_1 > \tau_2$, Adv decides that C has not been recently requested by anyone from its side of R.[7]

This simple timing attack is applicable to all ICN architectures that employ in-network caching. In NDN, however, it is even easier to discover the content of R's cache by exploiting the `Scope` and `Exclude` interest fields. Specifically, Adv can set the `Scope` field of an interest for R to "2". This value means that the interest can traverse at most two NDN entities, including the source. Thus, if Adv receives C for an interest with `Scope` equal to 2, then, regardless of the delay, C must have been satisfied from R's cache. To make matters worse, Adv is not even required to guess the name of C to discover its presence. Together, longest-prefix matching on content names and the `Scope` and `Exclude` fields can be used to implement an NDN-specific feature called *harvesting*. To show how this works, assume that Adv sends an interest to R with the name "/" and scope equal to 2. This ensures that R will satisfy the interest from its cache with a content object C_1 whose name is prefixed with "/". Since this criteria holds for *any* content with a valid name, R will always reply from its cache with some content object. Adv can then send another interest with the name "/" and scope equal to 2 which excludes C_1. This causes R to respond with another content from its cache $C_2 \neq C_1$. Adv can easily repeat this process indefinitely to receive a copy of all content in R's cache and, as a result, *harvest* that cache without doing any timing measurements. To prevent or mitigate this exploit, NDN routers are allowed to disregard the `Scope` and `Exclude` fields without significantly affecting any networking functionality. Therefore, we do not discuss harvesting further and only focus on time-based attacks.

To illustrate the ease of carrying out these timing attacks, we ran some experiments using the topology shown in Figure 1. It includes: (1) consumer Cr, (2) router R, (3) producer P, and (4) adversary Adv. P is reachable by Cr and Adv only through R. We also assume that only Cr and Adv are served by R as their fist-hop router, though we will later relax this assumption. In all of our experiments, P publishes 1000 content objects with different names. Moreover, the type and characteristics of the links connecting Cr, Adv, and P to R differ based on each experiment setup. We describe the detailed setup and experiment outcomes below.

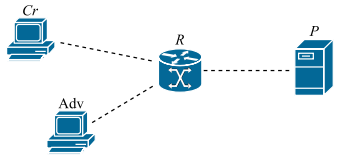


Fig. 1. Cache privacy attack experimental setup

Consumer Privacy in LAN Environment. In this environment, Cr, Adv and P are all connected to R via wired Ethernet. Cr starts by requesting all published content objects, which causes them to be cached by R. Then, Adv requests the same content objects which are promptly fetched from R's cache. We measure average delays for retrieving content from P and R. The results, illustrated in Figure 2(a), show the probability distribution function (PDF) for these delays. In this experiment, Adv can determine with probability greater than 0.99 whether C is retrieved from R's cache.

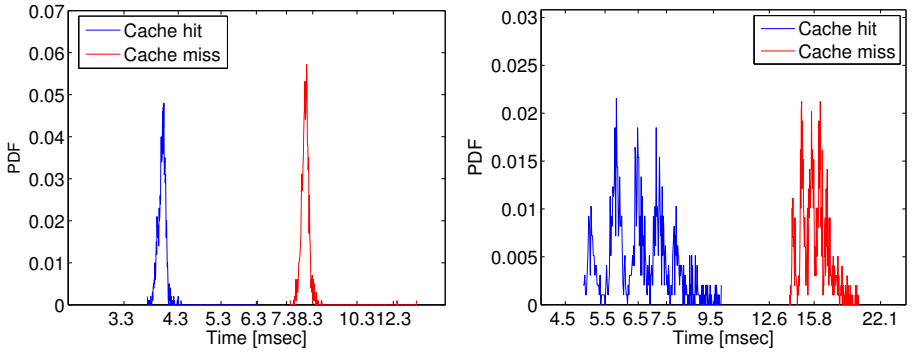
Consumer Privacy in WAN Environment. We also run similar experiments in a WAN topology using the NDN testbed [28]. In

5. Exceptions to this include nameless content objects, and content object authenticated via MACs.

6. In fact, NDN's codebase and design originated with an early version of CCN.

7. $\tau_1 < \tau_2$ may occur. Since $\tau_1 \approx \tau_2$, it may be the case that $\tau_1 < \tau_2$ due to time resolution errors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing



(a) Consumer privacy in LAN environment

(b) Consumer privacy in WAN environment

(c) Consumer privacy in local environment

(d) Producer privacy in WAN environment

Fig. 2. Timing Attack Results

this case, Cr and Adv both connect to the same first-hop router R' via Ethernet. R' is 3 hops away from R via a WAN link. Similarly, P is also 3 hops away from R over WAN. Figure 3 shows the setup of this experiment and Figure 2(b) shows the results. Clearly, the presence of additional hops increases delay and introduces some variance. However, Adv can still determine whether C is retrieved from R's cache with probability greater than 0.99.

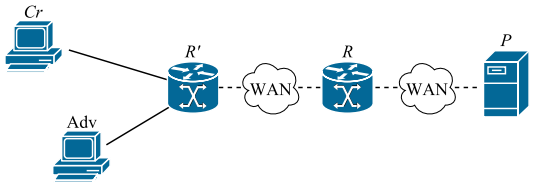


Fig. 3. Consumer privacy in WAN environment topology

Consumer Privacy in a Local Environment. The two attacks described thus far (in LAN and WAN environments) are also applicable to the local cache of a specific ICN node, e.g., a laptop or Android smartphone [7] with a cache shared among several applications, as shown in Figure 4. A malicious application can abuse this cache by employing the same probing techniques described above, hence learning which content was retrieved by a honest application running on the same host. Figure 2(c) summarizes our results in the local environment settings.

The difference between cache hits and cache misses is even more evident than in previous experiments, because the cost of retrieving content locally is significantly lower than the cost of retrieving content from a different node. In this setting, Adv learns information about cached content requested by other applications with overwhelming probability.

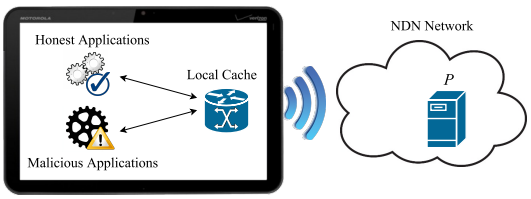


Fig. 4. Consumer privacy in local environment topology

Producer Privacy in WAN Environment. We now turn to producer's privacy as described in Section 1. We consider the topology in Figure 5, where P is directly connected to R, while Cr and Adv are three hops away. Adv's goal is to determine whether C was recently served by P and is cached by R.

Figure 2(d) summarizes the results of our experiments. In this setting, Adv can distinguish whether C is served from R with probability 0.59 by probing a single content object. However, as mentioned in Section 2, large pieces of content are typically split into several (smaller) pieces and transmitted as multiple content objects. In practice, the correlation between such objects could be exploited to improve Adv's cache privacy attack accuracy, because Adv only needs to determine whether *one* of the correlated content object pieces has been served by P. However, in the following analysis, we will assume that chunks of a partitioned content object are independent.

Let **success** denote the event where Adv successfully determines whether a single content object is fetched from the cache, and let **fail** denote failure to do so. Since **fail** and **success** are independent for all content objects, overall probability of failure (**FAIL**) can be expressed as:

$$\Pr[\textbf{FAIL}] = (\Pr[\textbf{fail}])^n$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

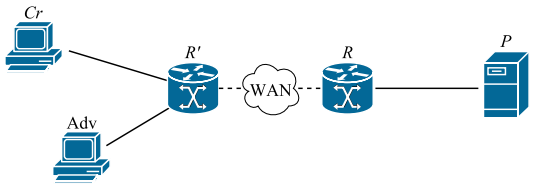


Fig. 5. Producer privacy in WAN environment topology

Similarly:

$$\Pr[\textbf{SUCCESS}] = 1 - (\Pr[\textbf{fail}])^n$$

In our experiment we have $\Pr[\textbf{success}] = 0.59$, and therefore $\Pr[\textbf{fail}] = 0.41$. If a large content is split into eight content objects, then the probability of a successful cache privacy attack can be increased to $\Pr[\textbf{SUCCESS}] = 1 - 0.41^8 \approx 0.999$

4 System, Adversary and Privacy Model

In this section we introduce our system, privacy, and adversary models.

4.1 System Model

Let Σ^* and Γ denote the universes of all names (some finite alphabet Σ) and content objects, respectively. As before, let R be an router. Internal state of R is represented by a function $S : \Gamma \to \Sigma^* \cup \{0\}$ that, for a given content object C in R's cache, represents the number of times C has been forwarded. $S(C) = 0$ for all C not in R's cache. We assume that C can appear in R's cache only if it has been previously forwarded by R.

We define a cache management algorithm $\mathcal{CM}$ that uses R's internal state to determine what content forwarded by R needs to be cached. $\mathcal{CM}$ also controls how R responds to interests that correspond to cached content. Without loss of generality, we assume that consumers have access to content only through R, i.e., R is their only choice as a first-hop router. To simplify the presentation of our results, we do not make any assumption on the size of R's cache, because cache size has only marginal impact on the results presented in the rest of the paper. We also make no assumptions about how $\mathcal{CM}$ responds to interests that match content in its cache, e.g., $\mathcal{CM}$ is free to ignore the cache altogether for some incoming interests. Finally, we assume that $\mathcal{CM}$ can hide cache hits (e.g., by simply not using its cache) but cannot hide cache misses.

By interacting with R, consumers are allowed to determine whether a specific content has been forwarded by R via probing attacks. We model this by an algorithm $Q_S : \Sigma^* \to \{0, 1\}$ with access to the router's internal state S. Q_S outputs 1 if cached content C matches the input name. Otherwise, Q_S outputs 0. After each invocation of Q_S, S transitions to S' such that $S'(C) = S(C)+1$ and, for all other $C' \neq C$, $S'(C') = S(C')$.

4.2 Adversary Model

The goal of Adv is to learn information about content forwarded by, and likely still cached in, R. Since $\mathcal{CM}$ is not secret, Adv can use Q_S to learn private information. In particular, Adv can test whether C was recently forwarded by querying $Q_S(n)$, for any $n \in \Sigma^*$.

In our model, Adv can be any ICN entity that requests and receives content. Adv is not allowed to compromise any honest (intended victim) consumers or R. Also, Adv cannot eavesdrop on communication between R and honest consumers. This restriction can be enforced in practice by using an encrypted channel between each consumer and its closest router.

Note that this model does not assume knowledge of any consumer adjacent to R. However, Adv's success in learning what content is cached in R can be used to help identify these consumers.

4.3 Privacy Model

We now turn to privacy definitions. We take advantage of the concept of (ε, δ)-probabilistic indistinguishability [17], [23] – a standard notion to measure indistinguishability of two distributions in privacy-oriented applications.

Definition 4.1 ((ε, δ)-probabilistic indistinguishability)**.** *Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are (ε, δ)-probabilistically indistinguishable, if we can divide the output space $\Omega = Range(\mathcal{D}_1) \cup Range(\mathcal{D}_2)$ into Ω_1 and Ω_2 such that:*

- *for all $O \in \Omega_1$, $e^{-\varepsilon} \leq \frac{\Pr(\mathcal{D}_1=O)}{\Pr(\mathcal{D}_2=O)} \leq e^{\varepsilon}$*
- $\Pr(\mathcal{D}_1 \in \Omega_2) + \Pr(\mathcal{D}_2 \in \Omega_2) \leq \delta$

Two distributions are "close" if both ε and δ are small. This definition is stronger than the widely used *statistical indistinguishability* since it requires similar probabilities for *each* output in Ω_1. Ω_2 contains all "bad" outputs with probabilities in $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ substantially; their ratios cannot be bounded by e^{ε} or even $e^{-\varepsilon}$. Intuitively, if $\mathcal{D}_1$ and $\mathcal{D}_2$ represent output distributions of $\mathcal{CM}$ with two different states, then (ε, δ) measures the information that $\mathcal{CM}$ leaks about those states. Any output from Ω_2 typically leaks "too much" information, e.g., occurrence of any $O \in \Omega_2$ such that $\Pr(Q_{S_1}(n) = O) > 0$ and $\Pr(Q_{S_2}(n) = O) = 0$, for the same name n and $S_1 \neq S_2$, may result in privacy breach in practice, as S_1 and S_2 become distinguishable through $\mathcal{CM}$ in that case.

We now define *perfect privacy* with respect to forwarded content. Informally, $\mathcal{CM}$ provides perfect privacy if the way it responds to Q_S queries does not yield any information about the content of R's cache.

Definition 4.2 (Perfect privacy)**.** *For all names $\ell \in \Sigma^*$, subset of content $M \subset \Gamma$, and pairs of states S_0, S_1 such that $S_0(x) = S_1(x)$ for all $x \in \Gamma \setminus M$ and $S_0(y) \neq S_1(y)$ for all $y \in M$, $Q_{S_0}(n)$ and $Q_{S_1}(n)$ are $(0, 0)$-probabilistically indistinguishable.*

The above definition is strong since it implies that no information is revealed about any content previously forwarded by R if $\mathcal{CM}$ offers perfect privacy. We believe that this level of privacy may not be necessary in practice. For this reason, we use the concept of content popularity to relax the above definition. Specifically, there is no need to conceal the presence of popular content objects (e.g., Google's home page) in router caches since Adv can safely assume that these objects are cached *without* probing them. To this end, let k be the number of requests after which a content is considered *popular*. We allow the distributions of Q outputs under two states S_0 and S_1 to be non-indistinguishable with some probability that depends on k.

Definition 4.3 ((k, ε, δ)-privacy)**.** *For all names $n \in \Sigma^*$, subset of content $M \subset \Gamma$, and pairs of states S_0, S_1 such that $S_0(x) = S_1(x)$ for all $x \in \Gamma \setminus M$ as well as $S_0(y) = 0$ and $0 < S_1(y) \leq k$ for all $y \in M$ (i.e., S_0 and S_1 only differ on content objects in M); $Q_{S_0}(n)$ and $Q_{S_1}(n)$ are (ε, δ)-probabilistically indistinguishable.*

5 Countermeasures

One trivial and effective countermeasure to the attacks in Section 3 is to simply disable router caching altogether. However,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

this would immediately worsen the performance of content distribution, one of the key features of ICN. In this section, we present a series of countermeasures against cache timing attacks. We discuss how ICN entities (particularly, routers) need to act upon encountering private content to prevent such attacks. We introduce techniques that inhibit Adv from extracting meaningful information about private content from router caches. These techniques provide various tradeoffs between privacy and performance.

In designing countermeasures, we consider two types of network traffic: *interactive* and *content distribution*. The first represents synchronous communication between two or more parties, e.g., voice/video conferencing and remote shell. This type of traffic is characterized by low-latency and continuous interaction, i.e., communicating parties continuously play the roles of both producer and consumer. Conversely, multimedia data delivery, live broadcasts, and delivery of web pages are examples of content distribution traffic. Our rationale for distinguishing between these two traffic types is discussed below.

5.1 Interactive Traffic

While in-network caching mostly benefits content distribution, it also helps mitigate packet loss in interactive communication [37]. This is because interests re-issued for lost packets can usually be satisfied by content cached closest to the location of the actual loss, thereby reducing delay for re-requested content. For this reason, any privacy-enhancing caching mechanism for this class of traffic should not introduce additional delay.

At the same time, since interactive content tends to be time-sensitive, there are hardly any benefits from caching it in routers in the longer term. For instance, if several users take part in a video-conference, cached stale video frames are of no use to any of them after a short amount of time.

We choose to protect this class of traffic using unpredictable names. Consumers and producers use a random value $rand$ as the last component of the name of each content they request and serve, respectively. This requires some coordination between the two (or more) parties involved in the interaction. Without loss of generality, the parties need to agree on a shared secret for seeding a pseudo-random function (e.g., a keyed cryptographic hash such as HMAC [21]) used to generate content-name-specific suffix $rand$.

We take advantage of our previous assumption that Adv cannot eavesdrop on consumers or producers involved in interactive communication or on traffic over R's incident links (e.g., due to link encryption or lack of physical access). Unpredictable content names inhibit malicious probing of R's cache. However, NDN routers must not respond with content that include $rand$ as a name component to interests that do not explicitly express it. For example, content named `/alice/skype/0/`$rand$ should not be returned to interests for `/alice/skype/` even though it would be a longest-prefix match. This is not an issue in CCN since the architecture uses exact content name matching. Moreover, in the event of packet loss, a consumer can re-issue an interest and still benefit from obtaining requested content from the router closest to the location of this loss.

5.2 Content Distribution Traffic

Unlike interactive traffic, content distribution does not require any coordination between producers and consumers. Also, it benefits a lot more from longer term router caching. Consequently, an ideal privacy approach for content distribution traffic would retain at least some benefits of caching beyond simple packet loss recovery.

In this setting, neither the consumers nor the timings of their requests are known in advance by the producer. Thus, using unpredictable content names, which seems well-suited for interactive traffic, is not viable for content distribution. We assume that privacy-sensitive content is identified either by interests carrying a single privacy bit or by the content itself carrying a similar bit. For the purposes of this type of traffic, we allow any combination of these markers.

Based on our discussion thus far, it appears that router involvement in handling private traffic is unavoidable for content distribution. Since low latency is not usually a primary requirement for this kind of traffic, routers can hide cache hits by introducing artificial delays before responding with privacy-sensitive cached content. Although this strategy increases end-to-end latency, it retains one important benefit of caching: reducing congestion and better bandwidth utilization. Moreover, if the overall delay introduced by routers is close to the RTT between Cr and P, the behavior of the network from Cr's perspective becomes similar to that of the current IP-based Internet.

We now need to consider *which* routers should introduce artificial delays. Since the objective of this countermeasure is to hide cache hits for privacy-sensitive content, it makes sense that only caching routers should introduce artificial delays. An interest might generate a cache hit in, *at most*, one router on the path between the consumer and the producer. Therefore, responding to interests should be delayed by, *at most*, one router – the router which satisfies the interest from its cache.

5.3 Artificial Delay Properties

Suppose that we introduce a constant delay γ such that, in case of a cache hit, R waits for γ before returning privacy-sensitive content. This procedure is shown in Algorithm 1. In case of a cache miss, the artificial delay at R must be the difference between γ and the actual delay for R to receive the requested content. Note that, in the latter case, the overall delay between the interest and content arrival times would still be γ.

Algorithm 1: Privacy-Aware-Forwarding

```
1: Input: Interest for C, privacy bit b
2: if C ∉ CS then
3:     Forward interest based on local forwarding strategy
4: else
5:     if b = 1 then
6:         Wait for some artificial delay based on C
7:     Forward C to downstream interface
```

This approach is easy to implement and requires very little additional per-cache-entry state. However, it has a major drawback in that it either penalizes nearby content or sacrifices privacy for far-away content. The former happens if γ is set too high and content with nearby consumers (with respect to R) becomes unduly delayed. The latter occurs when requested content is far away (or routed via slow and/or congested links) and the actual delay at R exceeds γ.

It is often impossible how to determine the optimal value of γ that would avoid both problems. Therefore, we consider two alternatives:

- *Content-specific delay:* For each privacy-sensitive content C, R stores the original interest-in $\rightarrow$ content-out delay γ_C. In other words, γ_C is the time it took R to obtain C, from either its producer or some other router's cache, the first time. If an interest for it arrives while C is in R's cache, R delays replying by γ_C.
- *Dynamic delay:* A router dynamically adjusts artificial delay to mimic current behavior of in-network caching for popular content. As the number of interests for a given content grows, so does the likelihood of it being cached at

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

a nearby router. According to Definition 4.2, artificial delay must not drop below actual delay for content located two hops from Adv. We describe this type of delay in Section 8.

The former is obviously the safer choice for privacy even though it imposes considerable delays for popular content that was originally fetched from far-away producers or routers and then cached at closer locations. Conversely, dynamic delay is more responsive to ephemeral traffic patterns at the cost of requiring routers to constantly monitor delay and popularity for all content.

Furthermore, for content to be perceived as satisfied by its producer, all routers on the path should take cache privacy into consideration. Consider the topology in Figure 6, which contains two consumer Cr_1 and Cr_2, two routers R_1 and R_2, and producer P. Assume that both routers start with empty caches and C is a content object published by P. Also, assume that both R_1 and R_2 spend the same amount of time forwarding an interest or satisfying it from their cache. The average network link RTTs are shown in Figure 6, where μ_i is the RTT of link i.[8]

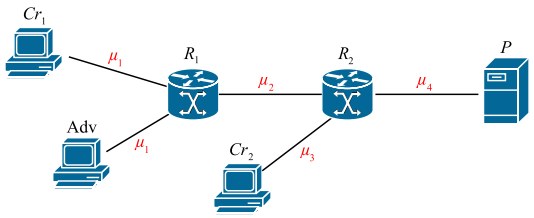


Fig. 6. Artificial delay length

We have the following three cases:

1) **Both routers introduce artificial delays when privacy-sensitive content is requested, content is requested from R_1.** If Cr_1 requests C, the latter is cached at both R_1 and R_2, consequently. Moreover, both routers also store the RTT to retrieve C from P. R_2 stores μ_4, and R_1 stores $(\mu_2 + \mu_4)$. If Adv requests C, R_1 delays with the already observed RTT $= (\mu_2 + \mu_4)$. In other words, RTT experienced by Adv would be $(\mu_1 + \mu_2 + \mu_4)$, which is the RTT to fetch C from P. Therefore, Adv is unable to determine if C has been previously requested.
2) **Both routers introduce artificial delays when privacy-sensitive content is requested, and content is requested from both R_1 and R_2.** If Cr_2 requests C, R_2 gets its retrieval RTT as μ_4. When Cr_1 requests C, R_2 adds an artificial delay of μ_4 before responding from its cache. This causes R_1 to store C's retrieval RTT as $(\mu_2 + \mu_4)$ when caching C. If Adv requests C, R_1 delays $(\mu_2 + \mu_4)$ causing Adv to experience RTT of $(\mu_1 + \mu_2 + \mu_4)$, which is also RTT to fetch C from P. Therefore, as in the previous case, Adv is unable to determine if C has been previously requested.
3) **Only R_1 is concerned about cache privacy, i.e., R_2 does not introduce any artificial delay when cache hits occur.** If Cr_2 requests C then this content gets cached in R_2. If Cr_1 requests C, R_1 caches it and stores its retrieval RTT $= \mu_2$. R_1 does not know that the request is satisfied from R_2's cache. Therefore, if Adv requests C, it experiences RTT $= (\mu_1 + \mu_2)$ due to the artificial delay introduced by R_1. This is different from the delay associated with retrieving C from P, and therefore Adv concludes that C has been requested before.

We thus conclude that if a router caches content then it should also introduce artificial delays. This requirement is stronger for edge routers and can be relaxed in the core due to their vastly larger anonymity sets.

5.4 Exceptions to Artificial Delay

We now consider how a router should handle all possible combinations of consumer- and producer-driven content marking. As mentioned above, if C is marked as private by its producer *or* any consumer, this must be always honored by routers as long as C is cached. However, this rule has exceptions, especially when specific content is consecutively requested as private and non-private by different (or even the same) consumers while cached by R. To demonstrate this, we use the topology in Figure 6 and assume that both routers are privacy-aware and start with empty caches.

If Cr_1 requests C for the first time with the interest privacy bit set, the content will be delivered from P and cached by R_1. Cr_1 experiences $(\mu_1 + \mu_2 + \mu_4)$ RTT delay. If Adv then requests C twice as non-private, it experiences the same (artificial) RTT delay of $(\mu_1 + \mu_2 + \mu_4)$ in both requests. This reveals the fact that a consumer previously requested C as private: if Adv was the only consumer to request C, the second request would have been satisfied with delay $2 \times \mu_1$ due to R_1's cache.

If Adv requests C as non-private, it is served by P and cached in R_1. First, after Adv's request, Cr_1 requests the same content as private, triggering R_1 to treat it as such in all subsequent requests. Also, stored delay will be $(\mu_2 + \mu_4)$ since that was the value recorded by R_1 when C was originally requested by Adv. Then, Adv requests C again as non-private. Adv sees an RTT delay of $(\mu_1 + \mu_2 + \mu_4)$. However, since Adv triggered caching C in its first request, this RTT delay is longer than expected. This again reveals that a consumer might have requested C as private in between two requests by Adv.

Although the previous two scenarios leak private information, Adv does not know whether privacy-sensitive content has actually been requested. This is because caching only happens for a limited time. In either scenario, Adv's second interest for C might have actually been satisfied by the producer because C was evicted from R_1's cache. However, with some additional information about R_1's traffic, i.e., knowledge about caching and eviction patterns at R_1, Adv can increase its success probability.

One way to address this information leakage problem is as follows: once an interest for C is marked as non-private, the content must be treated as non-private as long as it remains in a router's cache. We attempt to formally summarize the rules of router behavior when processing content requests.

1) Once a content is requested or served as non-private, it should always be treated as such as long as it is cached.
2) The effect of changing the privacy bit (or any other method of specifying content privacy) via an interest should always take effect after the interest is satisfied. Specifically, an artificial delay should not be used in response to the interest that sets the privacy bit. This delay should only be applied to all subsequent interests for the same content. This includes interests that are aggregated in the PIT with one which has the privacy bit set.

As mentioned in Section 7, requesting content as private encourages consumer selfishness. The general outcome would be detrimental for all consumers, who would experience high latency even when the requested content is cached. However, we claim that consumers have at least one incentive for requesting content without privacy: *reduced delay* for re-transmitted interests in case of packet loss.[9] Requesting content with privacy delays its re-transmission from router caches (if the original

8. The links Cr_1–R_1 and Adv–R_1 can have different RTT values. We chose to use the same one for demonstration simplicity.

9. Packet loss rate in today's Internet hovers around 4% [5].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

content object is lost) and hence results in higher delays. Thus, we believe that the rational choice for consumers is to request content with privacy only when actually needed.

6 Handling Distributed Adversaries

Techniques presented in Section 5 enable perfect-privacy in the presence of local adversaries. However, if the adversary is distributed and connected to multiple upstream routers, artificial delay introduced by the first-hop forwarder can be easily bypassed. This type of attack and its mitigation are addressed in this section.

6.1 Distributed Timing Attack

Consider the topology in Figure 6. In this scenario, Adv can retrieve content from both R_1 and R_2. We also assume that both routers honor private content requests and start with empty caches. The following attack shows how Adv can exploit access to R_1 and R_2 to circumvent the artificial delay countermeasure proposed above.

1) Cr_1 asks for C as private.
2) Adv asks for C first from R_1 and then from R_2 as non-private.

The delay for Cr_1's interest is $(\mu_1 + \mu_2 + \mu_4)$. The RTTs for Adv's interests are $(\mu_1+\mu_2+\mu_4)$ and $2\times(\mu_3+\mu_4)$, respectively. The latter is because Cr_1's interest marked the content as private in R_2's cache. If Cr_1 did not ask for this content, then the RTTs for both of Adv's interests would be $(\mu_1 + \mu_2 + \mu_4)$ and μ_3. Since neither Adv interest is private, the content is cached in both routers (as a consequence of Adv's first request) as non-private.

This attack works because R_1 and R_2 do not share content-specific privacy status bits. In other words, regardless of C's privacy status in R_1's cache, R_2 makes a local decision when receiving requests for C.

6.2 Mitigating Distributed Adversaries

We propose a simple technique to deal with this type of attack. Whenever a router makes a privacy status change for any cached content, it notifies the upstream router through which that content was received. We do not mandate a specific form of notification, as long as it contains the name of the content in question and some form of authentication. Each router treats these notifications as a request to change the local privacy status of the content in question, if it is still cached. The routers then forward the notification to their upstream peers. These steps are followed by routers along the path to the producer, until either the producer is reached, or a router which does not have the content in its cache is encountered. We expect that notifications would not travel far, because in practice most of the caching will likely occur at the edge of a network [13].

We illustrate how the proposed technique work using the example of distributed attack example described above. Assume that Cr_1 performs step (1) to obtain C from R_1 privately. If R_1 notifies R_2 about its privacy status change for C before replying to Adv, R_2 marks its cached copy of C as non-private. Observed RTTs from Adv's interests are $2 \times (\mu_1 + \mu_2 + \mu_4)$ and $2 \times \mu_3$, the same as if Cr_1 did not ask for C. Therefore, Adv cannot use these RTTs to determine whether Cr_1 asked for C.

The rationale behind this approach is as follows. Artificial delay works to prevent timing attacks because it hides the presence of cache content. If R_1 did not have a cache, Adv's first interest would be forwarded to R_2. In this case, Adv's timing attack would be is unsuccessful. Thus, the notification supplements artificial delay to emulate this scenario.

Also, using notifications and artificial delay is much more efficient than forwarding interests without delays, because notifications (as described) only carry a content name and do not expect a subsequent content response.

7 Which Content is Private?

When a content object is marked or requested as private, the network should provide (at least) (k, ε, δ)-privacy based on Definition 4.3. However, since not all content is private, the question remains: how does the network determine which content is private? Unfortunately, there is no universal policy. All three ICN entities (consumers, routers, and producers) must individually or collectively participate in this decision. In the rest of this section we present advantages and drawbacks of individual privacy decisions, along with the involvement level of each entity. We then discuss privacy decision techniques that require the collaboration of several network entities.

7.1 Router-Driven

Since cache management is part of a normal router's standard operating procedure, it might seem obvious that this router involvement in protecting cache privacy is required. However, as we show below, it is possible to achieve privacy without network (router) awareness. Routers solely deciding what content is private is problematic. This is because they do not have the means to differentiate between requested content. Any attempt in doing so increases overhead and negatively impacts on network throughput. Because routers cannot attempt to differentiate between requested content, achieving cache privacy requires treating all content as private, hence providing unnecessary perfect privacy as per Definition 4.2. This adds complexity and overhead to router operations, thus inhibiting efficient content distribution.

7.2 Consumer-Driven

In this case, consumers are responsible for specifying what content is private. When consumers connect to the network, they can specify whether their traffic should be treated as private. Both consumer and router involvement is required since the latter performs all cache management operations.

Communicating what content is private between consumers and routers can be done using one (or both) of the following methods:

1) Interest messages can carry a `NO-Cache` flag. When set, routers do not cache corresponding content objects. This clearly protect consumers privacy.
2) Interest messages can carry a `Privacy` bit. When set, corresponding content objects are marked accordingly and treated as private when cached by routers. The producer may or may not honor this `Privacy` bit.

The advantages of consumer-driven privacy decisions are: (1) finer granularity than routers treating all (or per-consumer) traffic as private, and (2) consumers know in advance whether requested content is private. However, this technique also suffers from the selfish consumer phenomena, e.g., consumers might always set `Privacy` bit in all interests.

7.3 Producer-Driven

In this case, producers specify which content is private. This approach requires producer and router involvement. Communicating content privacy to the network can be done using one of the methods mentioned above, i.e., content object header can include either a `NO-Cache` flag or a `Privacy` bit. A drawback of this technique is that consumers do not know in advance whether requested content will be treated as private.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

One difference between producer-driven and consumer-driven methods is that the former does not suffer from the "selfish producer" phenomena. Selfish behavior by producers is nonsensical since it defeats producers objective of efficient content delivery.

7.4 Collaborative Privacy Decisions

It is possible for both consumers and producers to determine the privacy level of a certain content. In fact, the consumer might set the `Privacy` bit in an interest and the producer does not set this bit in the corresponding content C. All routers caching this content should respect the `Privacy` bit, i.e., C is considered private if it is either requested or served as such.

Another alternative is to use unpredictable (secret) names for content. In other words, if C is private, both consumers and producers refer to it by a name that contains a random and a hard-to-guess component, ideally derived from some shared secret. One disadvantage of this approach is that both parties need to be involved and a priori agree on the random name component. On the other hand, an important advantage of this approach is its opaqueness since routers need not be involved. As discussed earlier, this approach is useful, and in fact recommended, for interactive traffic.

8 Improved Privacy-Utility Trade-Off

So far we considered techniques where cache hits for private content are always delayed. Proposed techniques are secure according to Definition 4.2, i.e., perfectly private, for all privacy-sensitive content. This is a strong security notion which may not be required in practice. We believe that there are factors, such as content popularity, that allow us to avoid hiding cache hits for private content without significantly compromising privacy. In this section, we discuss and analyze more practical techniques that relax the *perfect privacy* requirement in favor of better performance, i.e., higher utility. In general, such techniques randomly decide whether to mimic a cache hit or a cache miss for each content request. The distribution of observed output reflects the (local) popularity of requested content.

8.1 A Non-Private Naïve Approach

Let ρ_C denote the number of requests for particular content C. The algorithm always generates a cache miss, iff $\rho_C \leq k$, where k denotes the size of the anonymity set. A cache hit indicates that at least k requests have been generated for C.

This approach has a drawback of Adv being able to determine whether C was previously requested. To do so, Adv (knowing k) issues requests for C until it determines that the content is coming from R's cache. Let ρ'_C be the number of such requests. If $\rho'_C > 0$, Adv learns that exactly $k - \rho'_C$ requests have been issued for C.

8.2 Random-Cache

Security of the previous scheme depends on Adv's knowledge of k. Our next scheme – *Random-Cache* – selects a random k for each content. Thus, the index of the first cache hit in the output sequence is random, and should not leak information about the router's cache.

As shown in Algorithm 2, the scheme works as follows: the router maintains a counter ρ_C for each C. The first request for C is always a cache miss, and ρ_C is initialized to 0. Also, k_C is picked randomly from $[0, K)$ according to a distribution of domain $[0, K)$, described by a random variable $\mathbb{K}$. Upon receipt of a new request for C, the router increments ρ_C and checks whether $\rho_C \leq k_C$. If so, it generates a cache miss, and a cache hit otherwise.

Algorithm 2: Random-Caching

```
1:  Input: Request for content C, Domain size K, Distribution of 𝕂
2:  Output: Cache hit or cache miss
3:  𝕋 := set of received content
4:  if C ∉ 𝕋 then
5:      Select k_C from [0, K) with probability Pr(𝕂 = k_C)
6:      𝕋 := 𝕋 ∪ {C}
7:      ρ_C := 0
8:      Output cache miss
9:  else
10:     ρ_C := ρ_C + 1
11:     if ρ_C ≤ k_C then
12:         Output cache miss
13:     else
14:         Output cache hit
```

We define *utility* as the ratio of expected number of cache hits and the total number of requests for a given content, i.e., it represents the fraction of interests satisfied from the cache.

Definition 8.1 (Utility)**.** *Let $\mathcal{H}(\rho)$ denote the random variable describing the distribution of the number of cache hits depending on the total number of requests ρ ($\rho \geq 1$). The utility function $u : \mathbb{N} \to \mathbb{R}^+$ of a cache management scheme is defined as: $u(\rho) = \frac{1}{\rho}\mathbb{E}(\mathcal{H}(\rho))$.*

In practice, we derive u using the average number of cache misses, instead of cache hits, which is easier to compute. Let $\mathcal{M}(\rho)$ denote the random variable describing the distribution of the number of cache misses, based on the total number of requests ρ ($\rho \geq 1$). Then, $\mathcal{M}(\rho) + \mathcal{H}(\rho) = c$, and $u(\rho) = 1 - \frac{1}{\rho}\mathbb{E}(\mathcal{M}(\rho))$.

Specifically, for *Random-Cache*, we have:

$$\mathbb{E}(\mathcal{M}(\rho)) = \sum_{i=1}^{\rho} i \cdot \mathsf{Pr}(\mathbb{K} = i-1) + \sum_{i=\rho+1}^{K} \rho \cdot \mathsf{Pr}(\mathbb{K} = i-1), \quad \text{if } 1 \leq \rho < K \tag{1}$$

$\mathbb{K}$ influences both privacy and utility. If cache misses occur with overwhelming probability, then we obtain (almost) perfect privacy with nearly no utility.

Uniform-Random-Cache If $\mathbb{K}$ is uniform, then we obtain the best privacy among all distributions in terms of ε (which is 0). Also, as shown below, we can decrease δ (and improve privacy) by increasing K at the cost of degrading utility. We refer to this instantiation of Random-Cache as *Uniform-Random-Cache*.

Formally, let $\mathcal{U}(0, K)$ denote a discrete uniform random variable, i.e., $\mathsf{Pr}(\mathcal{U}(0, K) = r) = 1/K$, $0 \leq r < K$. *Uniform-Random-Cache* is an instantiation of Random-Cache (Algorithm 2) with $\mathbb{K} = \mathcal{U}(0, K)$.

Theorem 8.1 (Privacy)**.** *If all cached content is statistically independent, Uniform-Random-Cache is $(k, 0, \frac{2k}{K})$-private.*

Proof. Slightly abusing the notation, let $Q_0(\mathcal{C}, r)$ and $Q_1(\mathcal{C}, r)$ denote the output of Algorithm 2 in states S_0 and S_1, respectively, with C when $k_C = r$. (Recall that $S_0(C) = 0$ and $S_1(C) = x$, where $1 \leq x \leq k$). In addition, $Q_0^t(C, r)$ and $Q_1^t(C, r)$ denote the sequence of outputs obtained by executing Algorithm 2 with C consecutively t times, in states S_0 and S_1, respectively. Since all content is statistically independent: (1) it does not matter whether S_0 and S_1 differ in more than one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

content's count, and (2) Adv's best strategy is to request the same content multiple times in order to infer information about router state. Let $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ denote two random variables describing $Q_0^t(C,r)$ and $Q_1^t(C,r)$ when r is selected uniformly at random according to Line 5 of Algorithm 2.

We show that, for all content C, $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ are $(0, \frac{2x}{K})$-probabilistically indistinguishable. This implies that *Uniform-Random-Cache* is also $(0, \frac{2x}{K})$–probabilistically indistinguishable with any C, assuming all content is statistically independent.

The output of $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ is a sequence with length t consisting of two sub-sequences; the prefix, which is composed of consecutive cache misses (i.e., sequence of 0's), and the suffix with consecutive cache hits (i.e., a sequence of 1's).

We partition output space $\Omega = \mathit{Range}(\mathcal{Q}_0^t) \cup \mathit{Range}(\mathcal{Q}_1^t)$ into Ω_1, Ω_2 and Ω_3, for all t and C, as follows:

- $\Omega_1 = \mathit{Range}(\mathcal{Q}_1^t) \setminus \mathit{Range}(\mathcal{Q}_0^t)$: If $r \in [0, x)$, then all the t replies are cache hits in state S_1. However, this output can not appear with S_0 where the very first answer is always a cache miss (the router first needs to retrieve the content). Thus, $\nexists r'$ such that $Q_1^t(C,r) = Q_0^t(C,r')$.
- $\Omega_2 = \mathit{Range}(\mathcal{Q}_0^t) \cap \mathit{Range}(\mathcal{Q}_1^t)$: If $r \in [x, K-x)$, then $Q_1^t(C,r) = Q_0^t(C, r-x)$.
- $\Omega_3 = \mathit{Range}(\mathcal{Q}_0^t) \setminus \mathit{Range}(\mathcal{Q}_1^t)$: If $r \in [K-x, K)$, then the output with S_0 contains at least $K-x+1$ cache misses, which is not possible with S_1. Hence, $\nexists r'$ such that $Q_0^t(C,r) = Q_1^t(C,r')$.

For output $O \in \Omega$, let $\mathit{prefix}(O)$ denote prefix length of O (i.e., # cache misses in O). Since k_C is selected uniformly at random, for all $O \in \Omega_2$, $\Pr(\mathcal{Q}_0^t = O) = \Pr(k_C = \mathit{prefix}(O) - 1) = \Pr(k_C = \mathit{prefix}(O) + x - 1) = \Pr(\mathcal{Q}_1^t = O)$. Hence, $\varepsilon = 0$. Moreover, if $O \in \Omega_1 \cup \Omega_3$, $\Pr(\mathcal{Q}_0^t = O) + \Pr(\mathcal{Q}_1^t = O) = \frac{1}{K}$. Since $|\Omega_1 \cup \Omega_3| = 2x$, we obtain $\delta = \Pr(\mathcal{Q}_0^t \in \Omega_1 \cup \Omega_3) + \Pr(\mathcal{Q}_1^t \in \Omega_1 \cup \Omega_3) = \frac{2x}{K} \leq \frac{2k}{K}$. □

This theorem states that the probability that Adv can determine whether a content has been requested zero or k times is $2k/K$. This is because observing any outcome (hit/miss) which can occur in state S_0, but not in S_1, (or vice-versa) occurs with probability $2x/K$. The analysis also shows that perfect privacy can not be achieved if a cache hit can be generated, with non-zero probability.

Theorem 8.2 (Utility)**.** *For Uniform-Random-Cache,* $u(\rho) = 1 - \frac{1}{\rho}\mathbb{E}(\mathcal{M}(\rho))$*, where*

$$\mathbb{E}(\mathcal{M}(\rho)) = \rho\left(1 - \frac{\rho - 1}{2K}\right), \qquad \text{if } 1 \leq \rho < K$$

Proof. The theorem follows from Equation 1. If $1 \leq \rho < K$, we have:

$$\begin{aligned}
\mathbb{E}(\mathcal{M}(\rho)) &= \sum_{i=1}^{\rho} i \cdot \mathsf{Pr}(\mathbb{K} = i-1) + \sum_{i=\rho+1}^{K} \rho \cdot \mathsf{Pr}(\mathbb{K} = i-1) \\
&= \sum_{i=1}^{\rho} i \cdot \frac{1}{K} + \sum_{i=\rho+1}^{K} \rho \cdot \frac{1}{K} \\
&= \left(\frac{\rho(\rho+1)}{2}\right)\left(\frac{1}{K}\right) + (K-\rho)\left(\rho \cdot \frac{1}{K}\right) \\
&= \left(\frac{\rho(\rho+1)}{2K}\right) + \rho - \left(\frac{\rho^2}{K}\right) \\
&= \rho + \frac{\rho^2+\rho}{2K} - \frac{2\rho^2}{2K} \\
&= \rho - \frac{\rho^2-\rho}{2K} = \rho\left(1 - \frac{\rho-1}{2K}\right)
\end{aligned}$$

□

Theorems 8.1 and 8.2 show that, by increasing the size of domain K, resulting privacy increases at the cost of degraded utility.

Exponential-Random-Cache. One drawback of uniform distribution is that having only one parameter (K) gives limited flexibility for adjusting the privacy/utility trade-off. Hence, we also consider truncated geometric distribution as a candidate for $\mathbb{K}$. The shape of this truncated geometric distribution can be calibrated through an extra parameter other than K. Assigning exponentially larger probability to small values of k_C results in fewer cache misses on average, at the cost of additional privacy loss (ε will increase). The corresponding scheme is called *Exponential-Random-Cache*.

Consider a random variable $\mathcal{G}(\alpha)$ with geometric distribution, i.e., $\mathsf{Pr}(\mathcal{G}(\alpha) = k) = (1-\alpha)\cdot\alpha^k$, where $k \geq 0$ and $0 < \alpha \leq 1$. Its truncated counterpart denoted by $\widetilde{\mathcal{G}}(\alpha, x_1, x_2)$ has a conditional probability distribution defined as: $P(\widetilde{\mathcal{G}}(\alpha, x_1, x_2) = k) = \frac{\mathsf{Pr}(\mathcal{G}(\alpha)=k)}{\sum_{i=x_1}^{x_2} \mathsf{Pr}(\mathcal{G}(\alpha)=i)}$ if $x_1 \leq k \leq x_2$ and 0 otherwise, where $[x_1, x_2]$ is the truncation interval ($x_1, x_2 \in \mathbb{N}^0$). Therefore:

$$\mathsf{Pr}(\widetilde{\mathcal{G}}(\alpha, 0, K) = r) = \frac{(1-\alpha)\cdot\alpha^r}{1-\alpha^{K+1}} \tag{2}$$

Exponential-Random-Cache is an instantiation of *Random-Cache* (Algorithm 2) with $\mathbb{K} = \widetilde{\mathcal{G}}(\alpha, 0, K-1)$. Here, α and K are input parameters of the algorithm that can be calibrated to achieve the desired privacy/utility trade-off:

Theorem 8.3 (Privacy)**.** *If all cached content is statistically independent, Exponential-Random-Cache is* $(k, -k\ln(\alpha), \frac{1-\alpha^k+\alpha^{K-k}-\alpha^K}{1-\alpha^K})$*-private*

Proof. The proof is similar to that of Theorem 8.1. We assume the same annotations and show that, for all content C, $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ are $(-k\ln(\alpha), \frac{1-\alpha^k+\alpha^{K-k}-\alpha^K}{1-\alpha^K})$-probabilistically indistinguishable

We identically partition Ω into Ω_1, Ω_2, Ω_3 similar to Theorem 8.1. If $O \in \Omega_2$,

$$\begin{aligned}
\frac{\Pr(\mathcal{Q}_0^t = O)}{\Pr(\mathcal{Q}_1^t = O)} &= \frac{\Pr(k_{\mathcal{C}} = \mathit{prefix}(O) - 1)}{\Pr(k_{\mathcal{C}} = \mathit{prefix}(O) + x - 1)} \\
&= \frac{\Pr(\widetilde{\mathcal{G}}(\alpha, 0, K-1) = \mathit{prefix}(O) - 1)}{\Pr(\widetilde{\mathcal{G}}(\alpha, 0, K-1) = \mathit{prefix}(O) + x - 1)} \\
&= \frac{\frac{(1-\alpha)\cdot\alpha^{(\mathit{prefix}(O)-1)}}{1-\alpha^K}}{\frac{(1-\alpha)\cdot\alpha^{(\mathit{prefix}(O)+x-1)}}{1-\alpha^K}} \\
&= \frac{\alpha^{(\mathit{prefix}(O)-1)}}{\alpha^{(\mathit{prefix}(O)+x-1)}} \\
&= \alpha^{-x}
\end{aligned}$$

Similarly, $\frac{\Pr(\mathcal{Q}_1^t=O)}{\Pr(\mathcal{Q}_0^t=O)} = \alpha^x$. Since $\alpha < 1$, we have $\varepsilon \leq \ln\alpha^{-k} = -k\ln(\alpha)$. In addition, $\Pr(\mathcal{Q}_1^t \in \Omega_1) = \sum_{i=1}^{x} \frac{(1-\alpha)\alpha^{i-1}}{1-\alpha^K} = \frac{1-\alpha^x}{1-\alpha^K}$, and $\Pr(\mathcal{Q}_0^t \in \Omega_3) = \sum_{i=K-x+1}^{K} \frac{(1-\alpha)\alpha^{i-1}}{1-\alpha^K} = \frac{\alpha^{K-x}-\alpha^K}{1-\alpha^K}$. Since $\Pr(\mathcal{Q}_0^t \in \Omega_1) = \Pr(\mathcal{Q}_1^t \in \Omega_3) = 0$, we get $\Pr(\mathcal{Q}_0^t \in \Omega_1 \cup \Omega_3) + \Pr(\mathcal{Q}_1^t \in \Omega_1 \cup \Omega_3) \leq \frac{1-\alpha^k+\alpha^{K-k}-\alpha^K}{1-\alpha^K}$. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

Theorem 8.4 (Utility)**.** *For Exponential-Random-Cache,* $u(\rho) = 1 - \frac{1}{\rho}\mathbb{E}(\mathcal{M}(\rho))$*, where*

$$\mathbb{E}(\mathcal{M}(\rho)) = \frac{1-\alpha^\rho-\rho\alpha^K}{1-\alpha^K} + \frac{\alpha(1-\alpha^\rho)}{(1-\alpha^K)(1-\alpha)},$$
$$\text{if } 1 \le \rho < K$$

Proof. Using the fact that $\sum_{i=1}^{\rho} i\alpha^{i-1} = \frac{d}{d\alpha}\sum_{i=1}^{\rho}\alpha^i = \frac{1-(\rho+1)\alpha^\rho}{1-\alpha} + \frac{\alpha(1-\alpha^\rho)}{(1-\alpha)^2}$ and $\sum_{i=\rho+1}^{K}\alpha^{i-1} = \frac{\alpha^\rho-\alpha^K}{1-\alpha}$ the theorem follows from Equation 1. If $1 \le \rho < K$, and using the conditional probability in Equation 2, we have:

$$\begin{aligned}
\mathbb{E}(\mathcal{M}(\rho)) &= \sum_{i=1}^{\rho} i \cdot \mathsf{Pr}(\mathbb{K} = i-1) + \sum_{i=\rho+1}^{K} \rho \cdot \mathsf{Pr}(\mathbb{K} = i-1) \\
&= \sum_{i=1}^{\rho} i \cdot \frac{(1-\alpha)\cdot\alpha^{i-1}}{1-\alpha^K} + \sum_{i=\rho+1}^{K} \rho \cdot \frac{(1-\alpha)\cdot\alpha^{i-1}}{1-\alpha^K} \\
&= \frac{1-\alpha}{1-\alpha^K} \cdot \sum_{i=1}^{\rho} i\alpha^{i-1} + \frac{\rho(1-\alpha)}{1-\alpha^K} \cdot \sum_{i=\rho+1}^{K} \alpha^{i-1} \\
&= \left[\frac{1-\alpha}{1-\alpha^K} \cdot \left(\frac{1-(\rho+1)\alpha^\rho}{1-\alpha} + \frac{\alpha(1-\alpha^\rho)}{(1-\alpha)^2}\right)\right] \\
&\quad + \left[\frac{\rho(1-\alpha)}{1-\alpha^K} \cdot \left(\frac{\alpha^\rho-\alpha^K}{1-\alpha}\right)\right] \\
&= \frac{1-(\rho+1)\alpha^\rho}{1-\alpha^K} + \frac{\alpha(1-\alpha^\rho)}{(1-\alpha^K)(1-\alpha)} \\
&\quad + \frac{\rho(\alpha^\rho-\alpha^K)}{1-\alpha^K} \\
&= \frac{1-\alpha^\rho-\rho\alpha^K}{1-\alpha^K} + \frac{\alpha(1-\alpha^\rho)}{(1-\alpha^K)(1-\alpha)}
\end{aligned}$$

□

8.3 Comparison of Proposed Schemes

Increasing α in the Exponential-Random-Cache scheme results in better privacy (smaller ε). However, δ can not be made arbitrarily small and it is ultimately determined by α. In particular, $\delta = 1 - \alpha^k$ when $K = \infty$, which is the smallest possible δ. In contrast, δ of the uniform distribution can be arbitrarily decreased by sufficiently increasing K.

We compare the utility of proposed schemes in Figure 7. In Figure 7(a), we adjust the same value of δ, that is 0.05, for both schemes, and plot their utility for different values of k while varying ε. In Figure 7(b), we compute the maximum value of $\varepsilon = -\ln(1-\delta)$ for various combinations of δ and k, and plot the difference between the utility functions of the two schemes for varying δ. Both figures show that the exponential scheme exhibits up to 12% performance gain over the uniform one. Figure 7(a) also shows that both schemes achieve better utility as the number of requests grows.

8.4 Addressing Content Correlation

Random-Cache requires statistically independent content in the cache, which is a very strong assumption. Multiple content objects may share the same name prefix, and their access patterns could be strongly correlated. Under this assumption, Random-Cache as described above becomes insecure since it allows Adv to sample multiple points under different k. By requesting a large number of related content objects, as soon as Adv receives one without any delay, it learns that, with overwhelming probability, the whole set of content has been requested before.

To alleviate this problem, correlated content must be grouped together. Algorithm 2 can then be applied to these groups rather than to individual content, i.e., using a single counter ρ_C and value of k_C. Even the above extension can not be proven secure against all correlation-based attacks. In many cases, content correlation is even more subtle (e.g., semantically related content having different names such as linked webpages). This might be identified with appropriate background knowledge. As a possible countermeasure, content could be augmented with a content ID field. Producers would populate such field with identical values for correlated content. Routers could then determine how to handle such content by observing this field. However, a thorough analysis of these attacks and of the corresponding countermeasures is beyond the scope of this dissertation.

9 Experimental Evaluation

We now evaluate the actual impact of cache privacy techniques through simulations. We do not include the method that involves notification messages, since we believe caching will most likely take place at the edge [13]. Thus, notification messages will traverse only a few hops before being dropped.

We experiment using HTTP traffic traces collected by IRCache [18], which is part of the National Laboratory for Advanced Network Research (NLANR) project [29].[10] Traces were collected on September 1, 2007 (over a 24-hour period), on a Web proxy located at Research Triangle Park, North Carolina. These traces reflect activity of 185 users, for approximately 3.2 million requests distributed over various destinations. We randomly divide these requests into two sets: private and non-private. Then, we "replay" them with the following algorithms:

1) **No Privacy.** Routers use no privacy-preserving techniques.
2) **Always Delay Private Content.** For each request of a (cached) *private* content, the router always generates a cache miss, while for *non-private* cached content the response is always cache hit. This implements the basic protocol in Section 5.2.
3) **Uniform-Random-Cache/Exponential-Random-Cache.** Requests for cached *private* content are handled according to Algorithm 2. Requests for *non-private* cached content always result in a cache hit.

A router caches all content and removes elements from its cache according to the LRU policy. In case of a cache hit, the corresponding cache entry becomes "fresh" even if the response is delayed.

For algorithms that classify content into private and non-private, each incoming request is randomly marked as private with probabilities 0.05, 0.1, 0.2, and 0.4. Without loss of generality, we assume that all content is of the same size. We consider caches of size: 2,000, 4,000, 8,000, 16,000, and 32,000 units (content objects). Furthermore, as a baseline, we also run the same algorithms with the cache of infinite size.

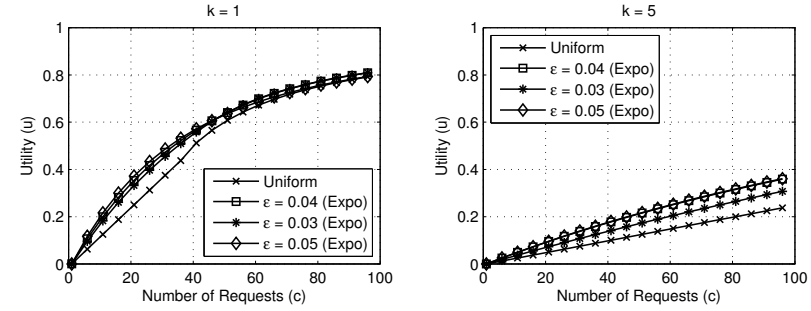
We set $k = 5$ and $\varepsilon = 0.005$. Results are plotted in Figure 8. Random caching algorithms have little impact on the percentage of cache hit rates. There is, at most, a 5% decrease in hit percentages for nearly all observed cache sizes. Increasing the amount of private interests also had little effect on the cache hit percentage. There is, at most, a 10% drop in the hit percentage as the percentage of private interests increased from 5% to 40%.

10 Bypassing Cache Delays

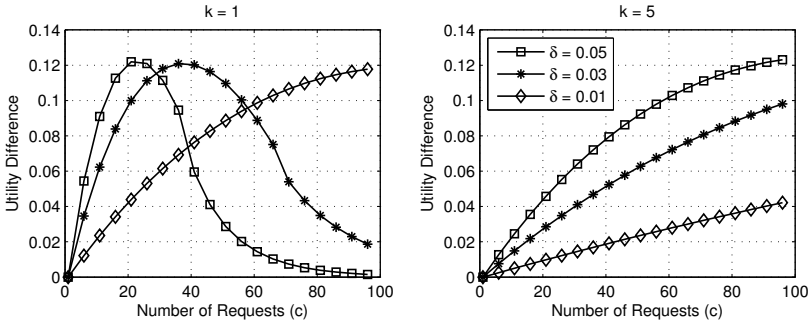
Thus far, our goal was to achieve perfect privacy, which we attained by introducing artificial delays in routers when cache

10. This was chosen since we believe it accurately reflects traffic that would be similar to NDN traffic. As of writing, there is no publicly available NDN traffic data set.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

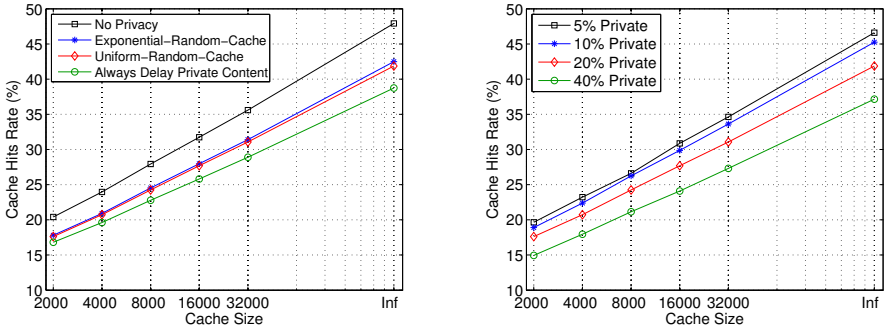


(a) Utility depending on privacy ($\delta = 0.05$)



(b) Maximal utility difference between *Uniform-Random-Cache* and *Exponential-Random-Cache* when $\varepsilon = -\ln(1-\delta)$

Fig. 7. *Uniform-Random-Cache* vs. *Exponential-Random-Cache*



(a) Comparison of our techniques

(b) *Exponential-Random-Cache* varying number of private requests

Fig. 8. Cache Hit Rates: Experimental Results

hits occur. We also described how to improve upon this deterministic delay with randomized delay algorithms. However, such strategies are problematic in certain scenarios:

1) If a consumer needs to resend an interest marked as private due to packet loss or transmission errors, the upstream router delays responding.
2) If a consumer is co-located with other trusted consumers then there is no reason the latter should be penalized by artificial delays if all parties request the same content. This is particularly true for the local application topology in Figure 4, where different applications running on the same consumer device may inherently trust one another since their common owner trusts all applications. In this scenario, if two applications request the same content through the local router, neither should be subjected to artificial delay.

In this section, we modify prior countermeasures so as to allow consumers to bypass the artificial delay. Our approach is based on the stateful techniques introduced in [24], [25]. It does not require any trust relationship between consumers and routers. The basic idea is that each consumer creates a special cache-bypass token $y = H(x)$ which is enclosed with each interest, where H is a cryptographic hash function and x is a random 256-bit string generated and stored for each request. The token is placed into a special per-hop header field called `Token`. When R receives an interest Int for C on interface F with a non-empty `Token` field, it performs the following steps:

1) If C is not cached locally, R forwards Int upstream according to its local forwarding strategy and stores `Token` in the PIT.
2) When C is returned, R stores the value of `Token` and F in the cache alongside C and its RTT.

If a consumer wishes to bypass the artificial delay for C, it must present the token pre-image x in an interest. The pre-image is also included in the `Token` field. Upon receipt of such an interest

from the same interface F, R performs the following steps:

1) R computes $y' = H(x)$.
2) If $y' = y$, R responds with C with no delays. Otherwise, R behaves according to the random cache algorithm described above.

This method does not introduce per-user state and requires no trust relationship between consumers and routers. However, one drawback is that routers are now required to store an additional 256-bit value for every cached content and perform one hash computation per interest.

The use of the bypass token is limited to consumers connected to the same local network (including different applications on the same device). The token can be exchanged between neighboring consumers using ARP-like request and response queries [31]. This, however, implies that Adv must not be able to eavesdrop on these messages, which complies with the adversary model in Section 4. However, Adv connecting to an encrypted network, e.g., in coffee shops with protected public access points, can still eavesdrop on all packets. In this case, there is no need for routers to obscure cache hits from cache misses since Adv learns what is requested by eavesdropping.

11 Related Work

Cache privacy in ICN, with NDN as a concrete example, was first studied in our work in [3] – the basis of the current paper. Mohaisen et al. [24], [25] study cache privacy in ICN (NDN) and propose countermeasures similar to ours. One primary difference is that their delay computation always samples from a normal distribution with parameters that change over time. In contrast, our work explores static uniform and exponential random delays. Moreover, Mohaisen et al. only consider privacy-preserving delays at edge routers. Proposed methods require keeping per-user state in routers in order to enable fast re-transmission of replies without artificial delays. In contrast, privacy information in our approach is distilled in a single bit and timestamp per content (and request). In our stateful variant described in Section 10, the state can service many consumers in the same subnet and thus has more value per bit. Also, [24], [25] does not consider distributed adversaries described in Section 6.

Outside of ICN, there is a large body of work on using side channels to extract information about other users' (or applications') behavior. Techniques proposed in [12], [16] allow malicious websites to learn whether a user visited a specific web page. The attacker sends a Java applet to the victim and detects cache hits with respect to the user's browsing cache.

Similarly, Felten et al. [11] show how a malicious website can determine whether a web page has been downloaded by its victim. The attack uses a Java applet or Javascript code and timing information to determine the content of the browser's cache.

Baron [4] proposes a countermeasure for attacks in [12], [16], based on completely hiding one's browsing history: rendering behavior of the browser (e.g., link colors, output of CSS functions) with respect to previously visited web pages is identical to that with new pages. However, this technique does not work for interactive attacks. In particular, Weinberg et al. [35] conducted experiments to show that interactive and timing attacks can still be used to disclose user's previous visited sites.

Bortz et al. [6] show two types of timing attacks that allow the adversary to learn the content of the browser's cache. The first, called *direct timing attack*, reveals whether one or more public websites have been visited by the victim. The second, *cross-site timing attack*, is more dangerous as it can reveal information about private sections of websites. For instance, it can determine whether a user is logged in to a specific service.

Another side-channel exploit is the timing attack on SIP VoIP networks. A tool described in [36] can be used to reveal the "calling history" of a SIP domain by observing which "recipient digital certificates" are stored in the local cache.

Several countermeasures to cache attacks have been developed. [20] proposes a server-side approach that prevents users from leaking the content of their cache. The idea is to randomize and personalize the links in web pages. Thus, a malicious site can not guess them when it tries to discover whether they have been visited.

Schinzel [32] discusses three techniques for mitigating timing-based side channel attacks in web applications. The first delays *all* responses such that the total delay of each response is identical. While this does not leak any information, it introduces considerable delay and affects user experience. The second approach entails adding a random delay to *each* response (the responses to identical requests are independently randomized). However, by requesting the *same* content sufficiently many times, the adversary can remove this random noise. The third approach is similar to the second. However, instead of randomizing the delay per response, a single random delay is selected per destination in order to prevent the aforementioned attack.

Timing and traffic analysis attacks have been extensively studied in the context of anonymity and mix networks such as Tor [26], [15], [33]. These attacks often attempt to reveal what particular servers or resources a client (consumer) requests. However, the mechanisms by which this is done is vastly different than which is proposed here. While our attacks (and countermeasures) depend solely on in-network caching, the Tor-related attacks rely on other anomalies such as producer delays, network delays due to Tor, etc.

Lauinger [22] considers several NDN-related security issues, identifies the problem of cache privacy and overviews several countermeasures, including some approaches similar to those in this paper. Crosby et al. [9] investigate how network latency deteriorates due to time-based side channel attacks, and design filters to reduce the effects of jitter.

12 Conclusion

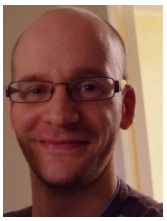
This paper explored cache privacy in ICN (and CCN) and identified several important privacy threats. We then introduced some plausible and effective counter-measures. First, we suggested that consumers and producers should indicate which content is privacy-sensitive. Then, we proposed several techniques that provide certain tradeoffs between privacy and latency. These techniques were assessed with respect to local and distributed adversaries. We also introduced a formal model that allows us to quantify the degree of privacy offered by various caching algorithms. We believe that proposed techniques are general and may be of interest beyond caching. Items of future work include analyzing analyzing the depth of edge routers which must introduce content-specific artificial delays as well as techniques for consumers and producers to link distinct private content together to prevent correlation attacks.

References

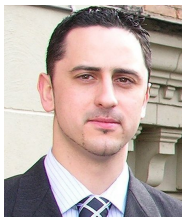
[1] CCNx 1.0 protocol specications roadmap. http://www.ietf.org/mail-archive/web/icnrg/current/pdfZyEQRE5tFS.pdf.
[2] Interest packet. http://named-data.net/doc/NDN-TLV/0.1/interest.html.
[3] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik. Cache privacy in named-data networking. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 41–51. IEEE, 2013.
[4] L. Baron. Preventing attacks on a users history through css: Visited selectors. *http://dbaron.org/mozilla/visited-privacy*, 2010.
[5] R. Birke, M. Mellia, M. Petracca, and D. Ross. Experiences of voip traffic monitoring in a commercial isp. *IJNM*, 20(5), 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2679711, IEEE Transactions on Dependable and Secure Computing

[6] A. Bortz and D. Boneh. Exposing private information by timing web applications. In *IW3C*, 2007.
[7] CCN now supports android. http://blogs.parc.com/blog/2010/11/ccn-now-supports-android//.
[8] CCNx Node Model. http://www.ccnx.org/releases/latest/doc/technical/CCNxProtocol.html.
[9] S. Crosby, D. Wallach, and R. Riedi. Opportunities and limits of remote timing attacks. *TISSEC*, 12(3), 2009.
[10] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *NDSS*, 2012.
[11] E. Felten and M. Schneider. Timing attacks on web privacy. In *CCS*, 2000.
[12] R. Focardi, R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, and E. Tronci. Formal models of timing attacks on web privacy. *ENTCS*, 62, 2002.
[13] J. Garcia-Luna-Aceves, A. Dabirmoghaddam, and M. Mirzazad-Barijoug. Understanding optimal caching and opportunistic caching at" the edge" of information-centric networks. In *Proceedings of the 1st international conference on Information-centric networking*, 2014.
[14] C. Ghali, G. Tsudik, and E. Uzun. Network-layer trust in named-data networking. *ACM SIGCOMM Computer Communication Review*, 44(5):12–19, 2014.
[15] Y. Gilad and A. Herzberg. Spying in the dark: Tcp and tor traffic analysis. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 100–119. Springer, 2012.
[16] R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, and E. Tronci. Automated analysis of timed security: A aase study on web privacy. *IJIS*, 2(3), 2004.
[17] M. Gotz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs–a comparative study of privacy guarantees. *IEEE TKDE*, 24(3):520 –532, 2012.
[18] IRCache Project. http://www.ircache.net/.
[19] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Co-NEXT*, 2009.
[20] M. Jakobsson and S. Stamm. Web camouflage: Protecting your clients from browser-sniffing attacks. *S&P Magazine*, 5(6), 2007.
[21] H. Krawczyk, R. Canetti, and M. Bellare. Hmac: Keyed-hashing for message authentication. 1997.
[22] T. Lauinger. Security & scalability of content-centric networking. Master's thesis, Technische Universitat Darmstadt, 2010.
[23] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.
[24] A. Mohaisen, H. Mekky, X. Zhang, H. Xie, and Y. Kim. Timing attacks on access privacy in information centric networks and countermeasures. 2015.
[25] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim. Protecting access privacy of cached contents in information centric networks. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 173–178. ACM, 2013.
[26] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 183–195. IEEE, 2005.
[27] Named Data Networking project (NDN). http://named-data.org.
[28] NDN Testbed. http://named-data.net/ndn-testbed/.
[29] The National Laboratory for Advanced Network Research Project. http://www.caida.org/projects/nlanr/.
[30] National Science Foundation of future Internet architecture (FIA) program. http://www.nets-fia.net/.
[31] D. Plummer. Ethernet address resolution protocol: Or converting network protocol addresses to 48. bit ethernet address for transmission on ethernet hardware. 1982.
[32] S. Schinzel. An efficient mitigation method for timing side channels on the Web. In *COSADE*, 2011.
[33] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*, pages 18–33. Springer, 2006.
[34] G. Tsudik, E. Uzun, and C. A. Wood. Ac3n: Anonymous communication in content-centric networking. In *CCNC*, 2016.
[35] Z. Weinberg, E. Chen, P. Jayaraman, and C. Jackson. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks. In *Symposium on S&P*, 2011.
[36] G. Zhang, S. Fischer-Hübner, L. Martucci, and S. Ehlert. Revealing the calling history of SIP VoIP systems by timing attacks. In *ARES*, 2009.
[37] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (NDN) project. *Technical Report NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.



Gergely Acs is a freelance researcher. His research focuses on different aspects of data privacy and security including privacy-preserving machine learning, data anonymization, and privacy risk analysis. Between 2010 and 2016, he was a research scholar and engineer at INRIA where this current work was also performed. He received his B.S., M.S., and Ph.D. degrees from the Budapest University of Technology and Economics, Hungary.

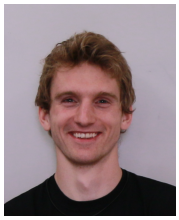


Mauro Conti is an an Associate Professor at the University of Padua, Italy. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as Assistant Professor the University of Padua, where he became Associate Professor in 2015. He was a Visiting Researcher at GMU (2008), UCLA (2010), UCI (2012, 2013, and 2014), and TU Darmstadt (2013). He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His main research interest is in the area of security and privacy. He is Associate Editor for several journals, including IEEE Communications Surveys & Tutorials and IEEE Transactions on Information

Paolo Gasti is an assistant professor of Computer Science at the New York Institute of Technology (NYIT), School of Engineering and Computing Sciences. Dr. Gasti's research focuses on behavioral biometrics, privacy-preserving biometric authentication and identification, secure multi-party protocols, and network security. Before joining NYIT, he worked as a research scholar at University of California, Irvine. His research has been sponsored by the National Science Foundation and the Defense Advanced Research Project Agency. He received his B.S., M.S., and Ph.D. degrees from University of Genoa, Italy. He is a Fulbright scholar, and member of the IEEE. He directs NYIT's Laboratory for behavioral Authentication, Machine learning and Privacy (LAMP).

Cesar Ghali Cesar Ghali received his Ph.D. from the University of California, Irvine majoring in Network Systems. He received his M.S. degree in Electrical and Computer Engineering at the American University of Beirut in 2010, where he also worked as a research assistant from 2008 till 2012. Before that, Cesar received his B.S. degree in Electrical Engineering from the University of Aleppo in 2007. Cesar's research interests include information security, network security and privacy, and web services and cloud computing security.

Gene Tsudik Gene Tsudik is a Chancellor's Professor of Computer Science at UC Irvine (UCI). He obtained his Ph.D. in Computer Science from USC in 1991. He began his research career at IBM Zurich Research Laboratory (1991-1996), followed by USC/ISI (1996-2000) and UCI (since 2000). His research interests include(d) numerous topics in security, privacy and applied cryptography. Between 2009 and 2016 Gene served as the Editor-in-Chief of ACM Transactions on Information and Systems Security (TISSEC). He's a former Fulbright Scholar, current Fulbright Specialist, Fellow of the IEEE, Fellow of the ACM and member of Academia Europaea. He directs UCI Secure Computing and Networking Center (SCONCE).



Christopher A. Wood is a fourth year Ph.D. student at the University of California Irvine, focusing on the intersection of content-centric networking security and privacy. He obtained a B.S. in Software Engineering and Computer Science and an M.S. in Computer Science from the Rochester Institute of Technology (RIT) in 2013. He was a member of the CCNx team at PARC between 2013 and 2016. Earlier, he interned at Intel, L-3 Communications, and other small software firms. Christopher is a recipient of the NSF GRFP fellowship, and a student member of the IEEE, SIAM, ACM, and IACR.